Button

Button The Button is one of the most commonly used fundamental components of AAVA Play. It is not just a click target. It's the pulse of the interface—alive, intentional, and radiant with purpose. The <aava-button> component provides a highly customizable clickable element with advanced visual effects including glass morphism, multiple interaction states, and comprehensive theming options. It supports various variants, sizes, icons, and custom styling while maintaining accessibility standards. How to use ■ import { AavaButtonComponent } from "@aava/play-core" ; Interactive Matrix ■ Explore all button combinations with our interactive playground. Basic Usage ■ Basic button implementation with default settings. Variants ■ The button component supports 9 semantic variants that control the visual appearance and meaning of the button. Available Variants ■ primary - Main call-to-action button (pink/brand color) secondary - Outlined style with transparent background success - Positive actions (green) warning - Cautionary actions (orange) danger - Destructive actions (red) info - Informational actions (blue) tertiary - Text-only action (transparent) Sizes ■ Five size options to fit different layout requirements and visual hierarchies. Available Sizes ■ xs (Extra Small) - Extra compact size for very dense interfaces sm (Small) - Compact size for dense interfaces md (Medium) - Standard size for most use cases (default) lg (Large) - Prominent size for primary actions xl (Extra Large) - Extra large size for hero sections and CTAs Hover Effects ■ Dynamic hover effects that enhance user interaction feedback. Available Hover Effects ■ torch - Internal semicircular sunrise effect (default) glow - Outer glow with elevation tint - Color overlay with brightness increase scale - Scale transformation with elevation Pressed Effects ■ Visual feedback for button press interactions with various animation styles. Available Pressed Effects ■ ripple - Multi-layered ripple animation (default) inset - Inset shadow effect Icons ■ Comprehensive icon support with flexible positioning and customization options. Icon Properties ■ iconName - Lucide icon name iconPosition - Position relative to text: left , right , icon-only iconColor - Custom icon color (defaults to currentColor) iconSize - Icon size in pixels (default: 20) States ■ Interactive states for different user scenarios and feedback. Available States ■ default - Programmatically active state processing - Loading/async operation state with pulse animation disabled - Non-interactive state Shapes & Styles ■ Shape modifiers and style variants for different design requirements. Available Shapes ■ default - Standard rectangular with border radius pill - Fully rounded corners (50px border radius) Style Variants ■ outlined - Transparent background with colored border clear - Transparent background with no border, uses variant text colors customStyles - Allows to apply inline CSS styles directly to the component.

This property accepts a key-value pair object where the key is the CSS property name and the value is the corresponding CSS value. Events ■ The button component emits events for user interactions. Available Events ■ userClick - Emitted on button click or keyboard activation (Enter/Space) Accessibility ■ The button component follows WAI-ARIA accessibility guidelines:

Proper keyboard navigation (Enter and Space key support) ARIA attributes for screen readers (aria-disabled , aria-pressed) Focus management with visible focus indicators Semantic button element usage API Reference ■ Inputs ■ Property Type Default Description label string '' Button text content variant ButtonVariant 'default' Visual variant: 'default' , 'primary' , 'secondary' , 'success' , 'warning' , 'danger' , 'info' size ButtonSize 'md' Button size: 'xs' , 'sm' , 'md' , 'lg' , 'xl' state ButtonState 'default' Interaction state: 'default' , 'hover' , 'active' , 'disabled' , 'processing' , 'focus' hoverEffect ButtonHoverEffect 'torch' Hover effect: 'torch' , 'glow' , 'tint' , 'scale' , 'none' pressedEffect ButtonPressedEffect 'ripple' Pressed effect: 'ripple' , 'inset' , 'solid' , 'none' processingEffect ButtonProcessingEffect 'pulse' Processing effect: 'pulse' , 'none' focusEffect ButtonFocusEffect 'border' Focus effect: 'border' , 'none' disabledEffect ButtonDisabledEffect 'dim' Disabled effect: 'dim' , 'none' disabled boolean false Whether button is disabled processing boolean false Whether button is in processing state pill boolean false Whether to use pill shape outlined boolean false Whether to use outlined style variant clear boolean false Whether to use clear style variant (transparent, no border) width string '' Custom width value height string '' Custom height value gradient string undefined Legacy – use customStyles instead background string undefined Legacy – use customStyles instead color string undefined Legacy – use customStyles instead dropdown boolean false Legacy – use separate dropdown component glassVariant ButtonGlassVariant 'glass-10' Default recommended glass intensity customStyles Record<string, string> {} CSS custom properties override iconName string '' Lucide icon name iconColor string '' Custom icon color iconSize number 20 Icon size in pixels iconPosition 'left' | 'right' | 'only' 'left' Icon position relative to text Outputs ■ Event Type Description userClick EventEmitter<Event> Emitted when button is clicked or activated via keyboard Design Tokens & Theming ■ AAVA Play buttons use semantic design tokens for all surfaces, spacing, radius, and motion. While global tokens define the visual language of the system, buttons expose a set of scoped override tokens that allow you to fine-tune appearance without breaking consistency. Use these only when necessary —for instance, to adjust a button's vertical padding inside a dense UI or to sharpen the radius for compact layouts. Available Design Tokens for Button ■ Size & Layout Tokens ■ Token Purpose Default Value --button-size-xsm-padding Padding for extra small size buttons Theme-based --button-size-sm-padding Padding for small size buttons Theme-based --button-size-md-padding Padding for medium size buttons Theme-based --button-size-lg-padding Padding for large size buttons Theme-based --button-size-xlg-padding Padding for extra large size buttons Theme-based --button-size-xsm-height Height for extra small size buttons Theme-based --button-size-sm-height Height for small size buttons Theme-based --button-size-md-height Height for medium size buttons Theme-based --button-size-lg-height Height for large size buttons Theme-based --button-size-xlg-height Height for extra large size buttons Theme-based --button-border-radius Border radius for default shape Theme-based Typography & Motion Tokens ■ Token Purpose Default Value --button-font-weight Font weight for button text Theme-based --button-transition Default transition animation Theme-based Icon Tokens ■ Token Purpose Default Value --button-icon-margin Margin around icons Theme-based Token Override Example ■ You can define overrides in your theme configuration or component styles: /* Custom

button theming */ .my-compact-buttons { --button-size-md-padding : 8 px 16 px ; --button-border-radius : 4 px ; --button-transition : 100 ms ease ; } This would make buttons more compact, sharper, and snappier—ideal for dense interfaces or admin tools. These tokens are opt-in overrides. If not set, the button will inherit global styles via the design system tokens. Best Practices ■ Design Guidelines ■ Use semantic variants - Choose variants that match the action's intent (primary for main actions, danger for destructive actions) Default variant - Use default variant for standard buttons; primary for main CTAs Size selection - Use medium as the default size; extra small / extra large for extreme cases only Effects system - Default effects work well together; customize only when needed Icon usage - Use icon-only for compact interfaces, left / right for labeled actions Consistent sizing - Match button sizes to surrounding elements and visual hierarchy Accessibility ■ Always provide meaningful labels - Even for icon-only buttons, use proper ARIA labels Keyboard navigation - Ensure all interactive elements are keyboard accessible Focus indicators - Maintain clear focus states for navigation Screen reader support - Use semantic HTML and proper ARIA attributes Color contrast - Ensure sufficient contrast for all variants and states Performance ■ Avoid excessive custom styling - Use built-in variants when possible Debounce rapid clicks - Prevent accidental multiple submissions Optimize icon loading - Use icon systems efficiently Consider bundle size - Import only needed effects and variants

```
<div class="matrix-demo">
  <div class="demo-header">
    <!-- Matrix Table -->
    <div class="matrix-table-container">
      <!-- Size Headers -->
      <div id="size-tabs" class="size-tabs">
        <aava-tabs
          [tabs]="sizeTabs"
          [activeTabId]="selectedSize"
          variant="button"
          size="sm"
          [showContentPanels]="false"
          (tabChange)="onSizeTabChange($event)"
          class="size-tabs-container"
        ></aava-tabs>
      </div>

      <!-- Matrix Grid -->
      <div class="matrix-grid">
        <!-- Column Headers -->
        <div class="matrix-header">
          <div class="mode-header">Mode</div>
          <div class="fill-header" *ngFor="let fill of fills">{{ fill }}</div>
        </div>

        <!-- Matrix Rows -->
        <div class="matrix-row" *ngFor="let mode of modes">
          <div class="mode-label">{{ mode }}</div>

          <div class="button-cell" *ngFor="let fill of fills">
            <ng-container
              *ngIf="getButtonConfig(mode, fill, selectedSize) as config"
            >
              <aava-button
                *ngIf="config.available"
                [label]="config.label"
                [pill]="config.pill"
                [outlined]="config.outlined"
                [clear]="config.clear"
                [size]="getSizeMapping(selectedSize)"
                [iconName]="config.iconName"
                [iconPosition]="config.iconPosition || 'left'"
                variant="primary"
                class="matrix-button"
              ></aava-button>

              <div *ngIf="!config.available" class="unavailable">
                ■ Not Available
              </div>
            </ng-container>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

```
---

type ButtonMode = 'pill' | 'default' | 'action' | 'quick-action';
type ButtonFill = 'filled' | 'outline' | 'clear';
type ButtonSize = 'xs' | 'sm' | 'md' | 'lg' | 'xl';

interface ButtonConfig {
  mode: ButtonMode;
  fill: ButtonFill;
  size: ButtonSize;
  pill: boolean;
  outlined: boolean;
  clear: boolean;
  iconPosition?: 'left' | 'right' | 'only';
  iconName: string;
  label: string;
  available: boolean;
  iconColor?: string;
}

modes: ButtonMode[] = ['pill', 'default', 'action', 'quick-action'];
  fills: ButtonFill[] = ['filled', 'outline', 'clear'];
  sizes: ButtonSize[] = ['xsmall', 'small', 'medium', 'large', 'xlarge'];
  sizeTabs: TabItem[] = [
    { id: 'xsmall', label: 'XSmall' },
    { id: 'small', label: 'Small' },
    { id: 'medium', label: 'Medium' },
    { id: 'large', label: 'Large' },
    { id: 'xlarge', label: 'XLarge' },
  ];
  variants: ButtonVariant[] = [
    'primary',
    'secondary',
    'success',
    'warning',
    'danger',
    'info',
  ];

  selectedSize: ButtonSize = 'md';
  selectedMode: ButtonMode = 'pill';
  selectedFill: ButtonFill = 'filled';

  toggleTheme(): void {
    this.isDarkTheme = !this.isDarkTheme;
    document.body.setAttribute(
      'data-theme',
      this.isDarkTheme ? 'dark' : 'light'
    );
  }

  onSizeTabChange(tab: TabItem): void {
    this.selectedSize = tab.id as ButtonSize;
  }

  getSizeLabel(size: ButtonSize): string {
```

```
  switch (size) {
    case 'xs':
      return 'XSmall';
    case 'sm':
      return 'Small';
    case 'md':
      return 'Medium';
    case 'lg':
      return 'Large';
    case 'xl':
      return 'XLarge';
    default:
      return size;
  }
}

getSizeMapping(
  size: ButtonSize
): 'xs' | 'sm' | 'md' | 'lg' | 'xl' {
  switch (size) {
    case 'xs':
      return 'xs';
    case 'sm':
      return 'sm';
    case 'md':
      return 'md';
    case 'lg':
      return 'lg';
    case 'xl':
      return 'xl';
    default:
      return 'md';
  }
}

getButtonConfig(
  mode: ButtonMode,
  fill: ButtonFill,
  size: ButtonSize
): ButtonConfig {
  // All fills are now available
  const baseConfig = {
    mode,
    fill,
    size,
    outlined: fill === 'outline',
    clear: fill === 'clear',
    available: true,
    iconColor: '#fff',
  };

  switch (mode) {
    case 'pill':
      return {
        ...baseConfig,
        pill: true,
```

```
        iconPosition: 'left' as const,
        iconName: 'star',
        label: 'Pill',
      };

    case 'default':
      return {
        ...baseConfig,
        pill: false,
        iconName: '',
        iconPosition: 'left' as const,
        label: 'Default',
      };

    case 'action':
      return {
        ...baseConfig,
        pill: false,
        iconPosition: 'left' as const,
        iconName: 'zap',
        label: 'Action',
      };

    case 'quick-action':
      return {
        ...baseConfig,
        pill: true,
        iconPosition: 'only' as const,
        iconName: 'plus',
        label: '',
      };

    default:
      return {
        ...baseConfig,
        pill: false,
        iconPosition: undefined,
        iconName: '',
        label: mode,
        available: false,
      };
  }
}
```

```
<aava-button
  label="Primary"
  variant="primary"
  (userClick)="onButtonClick($event)"
  [pill]="true"
>
</aava-button>
<aava-button
  label="Primary"
  variant="primary"
  (userClick)="onButtonClick($event)"
>
</aava-button>


---


  onButtonClick(event: Event) {
    console.log('Button clicked:', event);
  }



<aava-button label="Primary" variant="primary"></aava-button>
<aava-button label="Secondary" variant="secondary"></aava-button>
<aava-button label="Success" variant="success"></aava-button>
<aava-button label="Warning" variant="warning"></aava-button>
<aava-button label="Danger" variant="danger"></aava-button>
<aava-button label="Info" variant="info"></aava-button>
<avaa-button label="Tertiary" variant="tertiary"></avaa-button>



<aava-button label="Extra Small" variant="primary" size="xs"></aava-button>
<aava-button label="Small" variant="primary" size="sm"></aava-button>
<aava-button label="Medium" variant="primary" size="md"></aava-button>
<aava-button label="Large" variant="primary" size="lg"></aava-button>
<aava-button label="Extra Large" variant="primary" size="xl"></aava-button>
```

```
<aava-button
  label="Torch (Recommended)"
  variant="primary"
  hoverEffect="torch"
  [pill]="true"
>
</aava-button>
<aava-button
  label="Glow Effect"
  ariant="warning"
  hoverEffect="glow"
  [pill]="true"
>
</aava-button>
<aava-button
  label="Tint Effect"
  variant="success"
  hoverEffect="tint"
  [pill]="true"
>
</aava-button>
<aava-button
  label="Scale Effect"
  variant="danger"
  hoverEffect="scale"
  [pill]="true"
>
</aava-button>



<aava-button
  label="Ripple (Recommended)"
  variant="primary"
  pressedEffect="ripple"
  [pill]="true"
>
</aava-button>
<aava-button
  label="Inset Effect"
  variant="warning"
  pressedEffect="inset"
  [pill]="true"
>
</aava-button>
```

```
<aava-button
  label="Left Icon"
  iconName="star"
  iconPosition="left"
  variant="primary"
  iconColor="#fff"
>
</aava-button>
<aava-button
  label="Right Icon"
  iconName="star"
  iconPosition="right"
  variant="primary"
  iconColor="#fff"
>
</aava-button>
<aava-button
  iconName="star"
  iconPosition="icon-only"
  variant="primary"
  iconColor="#fff"
>
</aava-button>
```

■ No code found

■ No code found

■ No code found